

Data-Driven Discrete Planning for Targeted Hopping of Compliantly Actuated Robotic Legs

Daniel Seidel^{1,2}, Dominic Lakatos², and Alin Albu-Schäffer^{2,1}

Abstract—Motion planning for fast locomotion of compliantly actuated robotic legs is generally considered to be a challenging issue, posing considerable real-time problems. This is at least the case if time-continuous trajectories need to be generated online. In this paper we take advantage of a simple controller structure, which reduces the motion planning to a discrete-time planning problem, in which only a small set of input parameters need to be determined for each step. We show that for a planar leg with serial elastic actuation, hopping on a ground with stairs of irregular length and height can be planned online, based on a parameter mapping which has been learned in a data-driven manner by performing hopping trials with an adaptive exploration algorithm to evenly sample the parameter space. Experiments on a planar hopping leg prototype validate the approach.

I. INTRODUCTION

Compliant actuators in robotic legs can increase robustness, performance, and energetic efficiency [1], [2]. Introducing springs in the joints of the rigid multi-body system results in a dynamics which acts as a low-pass filter between the links and the motors, thus reducing force peaks due to impacts and unexpected contact forces. In particular, the capability of buffering and directed releasing elastic energy increases the available peak power at the links (w.r.t. the motors). By exploiting resonance like effects also the energy consumption can be reduced. This is of major importance especially for explosive motion tasks such as aperiodic jumping. On the other hand, introducing springs in the drive trains of the robotic joints also increases the order of the dynamics such that the control variables cannot be directly actuated. This makes the planning and control a challenging task.

There exist several approaches to plan and control highly dynamical periodic or explosive motions of compliantly actuated robots so far [3], [4], [5], [6], [7], [8], [9], [10], [11], [12]. The approaches of [3] and [4] exploit the elasticities by constrained numerical optimization. The resulting optimal control strategy strongly depends on the dynamics model of the plant and the computational cost of the approach explodes with the number of degrees of freedom. While the pioneering methods of *Raibert* [5] benefit from a harmonization of the robot and control design, the approaches [6], [7], [8], [9] are based on fundamental virtual and template models of dynamic locomotion. These models have to be implemented

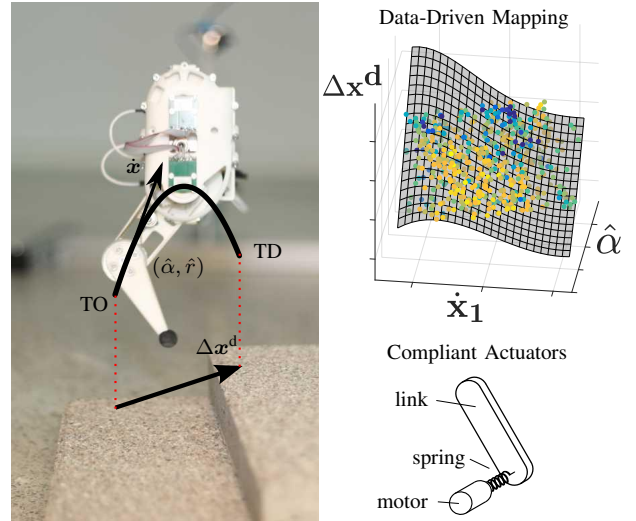


Fig. 1. Targeted hopping based on data-driven mapping of a compliantly actuated robotic leg. The input to the mapping is the trunk velocity at touchdown $\dot{x}^{TD} \in \mathbb{R}^2$ and the desired foot displacement $\Delta x_1^d \in \mathbb{R}^1$. The output are the controller parameters angular switching amplitude $\hat{\alpha}$ and radial switching amplitude \hat{r} (not depicted in the plot).

in the deviating dynamics of the compliantly actuated segmented legs. The methods of [10], [11], [12] either search for a limit cycle dynamics or for a locomotion trajectory by extensive open-loop optimization or reinforcement learning. Since these approaches are designed to work off-line, they cannot adapt to model uncertainties or environmental changes.

In our previous work we demonstrated the concept of regulating the velocity of a stationary limit cycle for a hopping motion on a compliantly actuated robotic leg [13]. The underlying bang-bang controller allows to plan motions in discrete space with only a small set of parameters. In contrast to the model-based or simulation data based methods of [5] or [14] we achieved stable jumping velocities with a simple velocity regulator that adjusts the controller parameter once per jumping cycle without model knowledge of the plant. We accomplished a reproducible sequence of aperiodic jumps on uneven terrain on the real robot using parameters optimized in simulation. However, model identification and validation is a time consuming process, which needs to be repeated for each prototype modification.

In this paper, we directly incorporate feedback of the plant in the planning process. By utilizing the feature of the discrete parameterization, a pure data-driven approach on parameter planning of directed aperiodic jumping motions is implemented, i.e., all action commands for the robot

¹ D. Seidel and A. Albu-Schäffer are with Technical University Munich, Chair of Sensor Based Robotic Systems and Intelligent Assistance Systems, Department of Informatics, D-85748 Garching, Germany daniel.seidel@tum.de

² D. Seidel, D. Lakatos and A. Albu-Schäffer are with the Robotic Mechatronic Center (RMC), Institute of Robotics and Mechatronics, German Aerospace Center (DLR), D-82234 Oberpfaffenhofen, Germany

are generated using data from previous real robot trials. The direct relation between the current state of the leg, the input parameters and a set of reference variables allows to generate a mapping for the control of the hopping motions, as depicted in Fig. 1. The intermediate step of parameter identification is not required, neither for the controller, nor for the planner. Moreover, the evaluation of all data recorded from the hardware system circumvents the need to search for parameters that produce a desired output, i.e., to compute the inverse mapping. This step is often done by means of extensive off-line optimization in simulation. In addition, we apply a simple but effective nearest-neighbor-based exploration approach to sample the input space. This allows to generate data for the mapping which covers all of the feasible input space while being able to choose the desired sampling density.

The paper is structured as follows: the idea of the pure data-driven planning approach is explained in Sect. II. Sect. III presents the underlying jumping control approach. The generation of the data-driven mapping is proposed in Sect. IV. Sect. V shows the performance of the approach. A brief conclusion is presented in Sect. VI.

II. THE IDEA

Pure data-driven approaches are not common in robotics applications, because it is typically expensive to generate a sufficient amount of training data. Problems of hardware wearout and usually no parallelism, especially when working with unique prototypes, often make large amounts of policy execution on hardware infeasible [15]. Furthermore, exploiting the configuration space of a robot to its limits may cause damage to the robot or make its use unsafe. For this reason simulations are often preferred, where execution is safe and parallelism is possible.

Nevertheless, for the following reasons we considered a pure data-driven approach to be feasible in the scenario of a planar leg as well as a quadrupedal robot in future experiments: (i) an underlying bang-bang controller reduces the parameter planning task to a time discrete problem, (ii) the use of serial elastic actuators prevents simple forward kinematics and dynamics calculations, (iii) since the controller does not require a dynamics model of the plant, the approach can be applied to low cost and rapid prototyping robotic legs.

By carrying out the parameter planning solely based on data recorded from hardware trials, the requirement to have an accurate system model and adapting it to every system change as well as to deal with deviations between simulation and real hardware is completely canceled. In our case, gathering new data to adapt control parameters is done within minutes. We plan to consecutively extend this method to more complex robotic systems, e.g. a quadrupedal robot, for which we are able to utilize a similar discrete control approach.

III. CONTROL OF JUMPING SEQUENCES

A. Control

This section presents the approach to control data-driven jumping sequences.

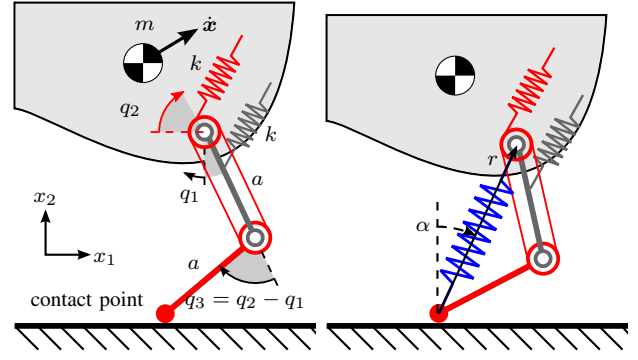


Fig. 2. Model of the compliantly actuated leg (left) and description of the stance dynamics in polar coordinates (right).

B. Model

The method will be introduced for the model of a compliantly actuated leg as shown in Fig. 2 (left). The leg is assumed to be attached to the main body (trunk) with very high inertial properties such that its rotation can be neglected, i.e., the trunk has only the two translational degrees of freedom of the plane.¹ The position of the trunk is measured by Cartesian coordinates $\mathbf{x} \in \mathbb{R}^2$. The thigh is connected to the trunk by a rotational joint with coordinate q_1 . The shank is hinged to the thigh with relative coordinate q_3 . There is a pulley concentric with the hip joint with relative coordinate q_2 , which couples to the knee joint such that $q_3 = q_2 - q_1$. The length of the leg segments is assumed to be equally $a > 0$ and the mass of thigh and shank is assumed to be negligible (compared to the mass of the trunk $m > 0$). Furthermore, each degree of freedom of the leg (thigh and pulley) is actuated via a linear spring with constant stiffness $k > 0$. A point-foot is considered, which is constrained during stance phases to touch the ground. This choice of model parameters has the advantageous implication that during the stance phase, the dynamics of the model expressed in polar coordinates (cf. Fig. 2 (right)) is governed

¹Note that this assumption holds especially for quadrupeds, where the fore- and hindlegs are configured symmetrically and the center of mass (COM) of the trunk is located at the center of pivot points of the legs.

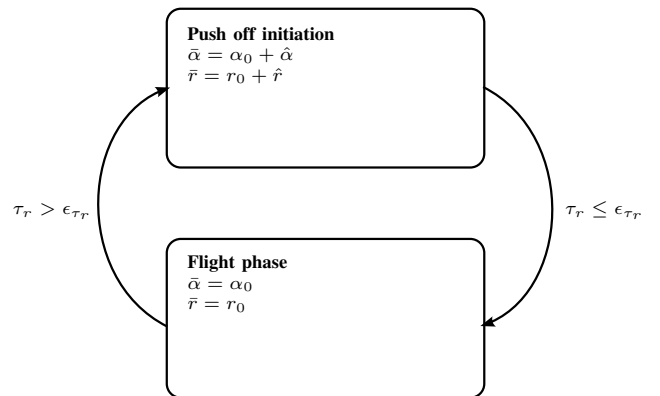


Fig. 3. Representation of the controller as finite state machine. The controller switches only between two constant positions.

by

$$m \left\{ \begin{bmatrix} r^2 & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \ddot{\alpha} \\ \ddot{r} \end{pmatrix} + r \begin{bmatrix} \dot{r} & \dot{\alpha} \\ -\dot{\alpha} & 0 \end{bmatrix} \begin{pmatrix} \dot{\alpha} \\ \dot{r} \end{pmatrix} + g_0 \begin{pmatrix} -\sin(\alpha)r \\ \cos(\alpha) \end{pmatrix} \right\} = -k \left(\frac{2[\alpha - \bar{\alpha}]}{\sqrt{4a^2 - r^2}} [\rho(r) - \rho(\bar{r})] \right). \quad (1)$$

Herein, α and r denote polar angle and radius, respectively, and $\rho(y) := \mp \arccos\left(1 - \frac{y^2}{2a^2}\right)$ represents an abbreviation related to the knee angle $q_3 = q_2 - q_1 = \mp \left[\arccos\left(1 - \frac{r^2}{2a^2}\right) - \pi \right]$.² The polar rest angle and length ($\bar{\alpha}, \bar{r}$) are considered as control input, i.e., the leg positions *before* the elastic component.

From (1) it can be seen that the dynamics of the considered articulated leg mechanism (in stance) behaves like the spring loaded inverted pendulum (SLIP) with an additional torsional spring acting in the direction of the polar angle. Therefore, the control method, as will be presented in the following, can be applied to all systems, which display such a fundamental template model behavior of legged locomotion.

The model (1) (capturing the dynamics during the stance phase) is conservative. However, the real system is subject to dissipation due to joint friction and impact losses. Therefore a controller is required, which counterbalances the energetic losses, and generates the (desired) jumping motion. The control principle is based on switching the rest positions of the springs [16]. It can be implemented by the finite state machine (FSM) as shown in Fig. 3. The FSM consists of one state for the flight phase and landing and one state for the pushoff initiation. The transition between these states is triggered, when the elastic force in radial direction

$$\tau_r = -k \frac{1}{\sqrt{4a^2 - r^2}} [\rho(r) - \rho(\bar{r})] \quad (2)$$

²The negative or positive sign selects the solutions $q_3 > 0$ or $q_3 < 0$, respectively.

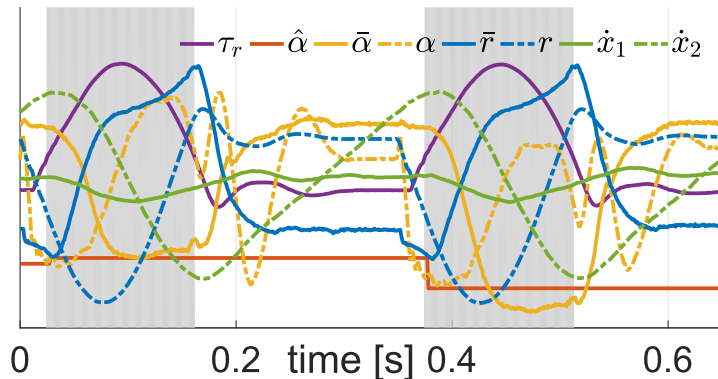


TABLE I
CONTROLLER PARAMETERS OF ONE JUMPING CYCLE.

steady-state orientation	α_0 in rad
steady-state length of leg axis	r_0 in m
angular switching amplitude	$\hat{\alpha}$ in rad
radial switching amplitude	\hat{r} in m
switching threshold	ϵ_{τ_r} in N

hits the threshold $\epsilon_{\tau_r} > 0$ from below (flight phase to pushoff) or from above (pushoff to flight phase). Thereby, the controller depends only on the parameters as listed in Tab. I and Fig. 4 (left) depicts measurements of all variables for a complete control cycle.

C. Discrete jump mapping

In the following, we define a mapping which relates a given initial state of the leg and a desired jumping distance to controller parameters which achieve this goal. On the basis of the events described in Fig. 5, we make simplifying assumptions on the controller parameters and states:

- 1) the configuration velocity of the leg at touchdown is negligible, i.e., $(\dot{\alpha}, \dot{r})^{\text{TD}} = \mathbf{0}$;
- 2) the configuration of the leg at touchdown is $\alpha = \bar{\alpha} = \alpha_0 = \text{const.}$ and $r = \bar{r} = r_0 = \text{const.}$;
- 3) the spring compression which triggers the switchings is constant, i.e., $\epsilon_{\tau_r} = \text{const.}$

Effectively, assumptions 1) and 2) mean that the leg reaches the steady state while in mid-air and the setpoint is defined by α_0 and r_0 . Additionally, we experimented with including parameter history of previous steps into the mapping, which helped to improve performance of the mapping. Then, given a desired jumping distance Δx_1^d , the discrete jump mapping

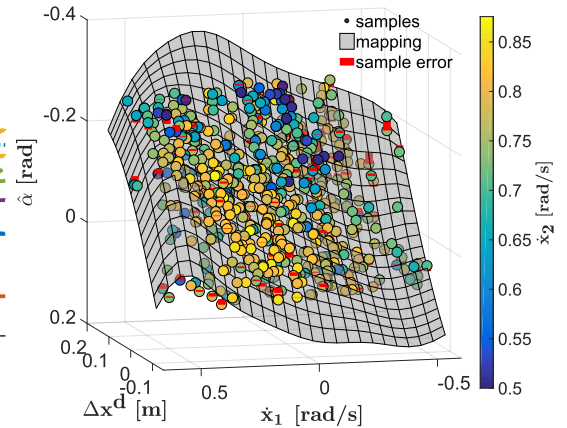


Fig. 4. Left: Plot of log data during jumping. Depicted is one and a half jump cycles. The area with grey background is the phase with ground contact as measured using τ_r . As this is threshold-based, one can see that there is a small delay between actual ground contact – τ_r starts increasing – and TD detection. The plot shows the behavior of the motor positions in polar coordinates $\bar{\alpha}$ and \bar{r} , link side positions α and r and trunk velocities \dot{x} . Values are scaled to fit the plot. Right: A full set of data points for which $\hat{\alpha}$ is plotted against a (desired) jump distance Δx_1^d and the horizontal velocity \dot{x}_1 . The third input dimension, the vertical velocity \dot{x}_2 , is coded by color. The grey surface is the resulting inverse mapping for generating control parameters. Depicted is a slice of the mapping calculated with mean vertical velocity of the data set. The plot shows the non-linear nature of the relation between inputs and outputs.

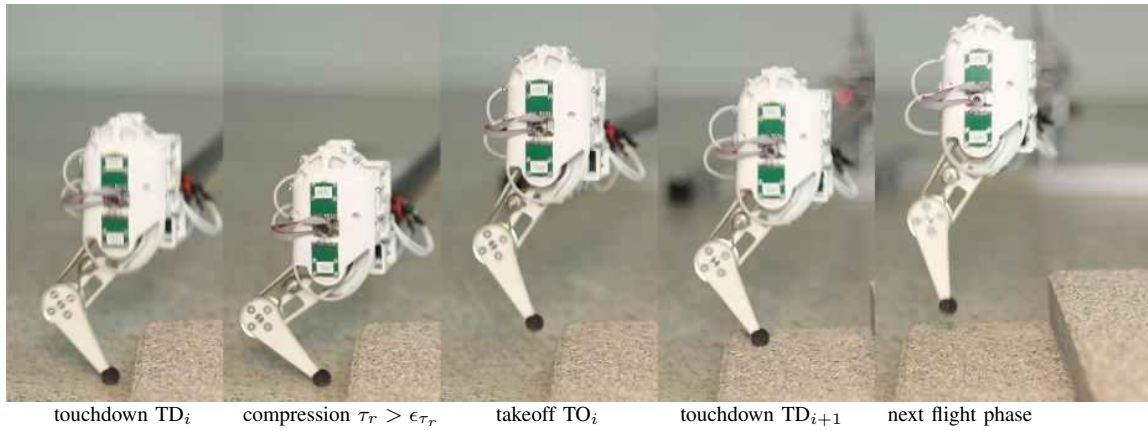


Fig. 5. Events of one jumping cycle used to define discrete jump mapping.

takes the form

$$\hat{\alpha}_t = \gamma(\Delta x_1^d, \dot{x}^{TD}, \hat{\alpha}_{t-1}, \zeta = \text{const.}) \quad (3)$$

where $\zeta = (\alpha_0, r_0, \hat{r}, \epsilon_{\tau_r})$ are parameters of the mapping.

IV. DATA-DRIVEN MAPPING GENERATION

In this section we describe the process of generating a mapping between input parameters of the controller and the system output from trials on the hardware. The goal is to produce input commands for consecutive experiments. This method exploits the advantage provided by the time discrete parameterization of the controller to easily generate data samples that correlate input parameters, input states and output states for each jump. Therefore, we first introduce the procedure for recording data samples. Based on this, we apply a regression model, namely Kernel Ridge Regression (KRR), to calculate a mapping that generates command inputs for desired outputs.

A. Data Acquisition

To effectively cover the state space, one has two options: sampling in the output space or in the input space. The first allows precise exploration of the desired behavior, however requires to use the imprecise backward mapping for "guessing" the appropriate inputs, i.e., online learning. Sampling directly in the input space has an uncertainty on the resulting output, but allows batch learning after data acquisition. We decided to take the latter approach, while a comparison of sampling effectiveness is planned for future work.

The data to calculate the mapping is generated by giving input commands to the robot and recording the response of the system. To be able to use the mapping universally for periodic as well as aperiodic jumping, the parameter sampling has to cover the input space as exhaustively as possible. While the control parameter part of the mapping can easily be sampled over the full range of the parameter, the system state part of the mapping cannot be commanded directly. Hence, the exploration has to generate motions that allow to sample the system state space as well. For our system this means that we need to generate samples that

result in output velocities which cover the full value range of the system.

Specifically, we have to enforce samples of three different types:

- Corresponding velocities and control parameters. These are samples which lie on the diagonal of an $\hat{\alpha}$ - \dot{x} -plot (as in Fig. 6). The control parameter is similar to the one that generated the input state.
- Higher input velocity than the output velocity with the next control parameter. Samples lie below the diagonal, the whole system is decelerated.
- Lower input velocity than the result with the next control parameter. Samples are above the diagonal and the system is accelerated.

In practice, extreme cases of the latter two types are abruptly jumping on the spot after a wide jump or even jumping backwards and vice versa.

The input and output parameters are either recorded as they are commanded to the leg (controller parameters ξ) or measured using sensors integrated in the plant. Motor positions are measured inside the servo motors and then transformed to polar coordinates $\bar{\alpha}$ and \bar{r} . The link side joint angles q (and therefore the leg positions in polar coordinates α and r) are measured with separate sensors after the spring element. The torques are estimated from the difference between motor and link position. The horizontal and vertical positions x are measured using angular sensors in the boom holding the trunk.

B. Exploration

We implemented a nearest-neighbor-based approach that calculates for a given input system state \dot{x}^{TD} and a predefined parameter range $[\hat{\alpha}_{min}, \hat{\alpha}_{max}]$ (i.e., the range in which valid motions are generated) the optimal next $\hat{\alpha}$ with respect to the current state of the exploration. For our system, $\hat{\alpha}$ is physically limited for backwards jumps by knee ground contact. For forward jumps, too high values cause the system to become unable to reach the steady state position before next ground contact. The method calculates the Euclidean distance between the input state and every sample of the current data set (normalized to $[0, 1]$). The samples that fall

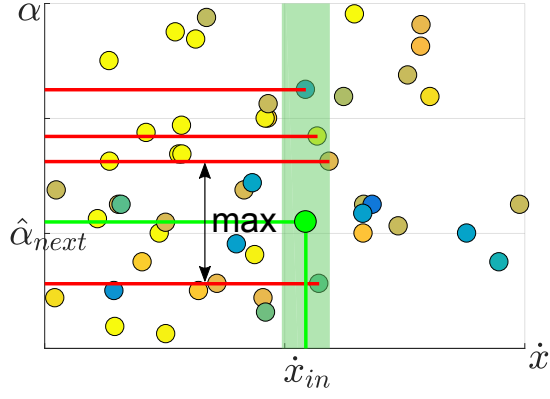


Fig. 6. 1D example for the exploration algorithm. For a given input velocity \dot{x}_{in} , all samples k with $\dot{x}_{in} - \epsilon_{\dot{x}} < \dot{x}_k < \dot{x}_{in} + \epsilon_{\dot{x}}$ are used to calculate $\hat{\alpha}_{next}$. It is chosen so that the distance to all other $\hat{\alpha}$ -values (including boundaries) is maximized. In the multidimensional case, the similarity is measured using the Euclidean norm of the difference between input vector and sample vector.

below a certain threshold $\epsilon_{\dot{x}}$ are considered as similar. The next $\hat{\alpha}$ is then chosen so that the difference to the set of all $\hat{\alpha}$ of the similar samples and the parameter limits is maximized (see Fig. 6). This allows to cover the entire state space with a small number of samples.

As a result, the method always chooses the region of the parameter space with least exploration for a given state and automatically samples the system state, since input velocities are always paired with a new control parameter, i.e., for similar input velocities the control parameter is always different. In consequence, a jump very likely results in an output velocity previously not measured and they are distributed over the whole state space. The density of the exploration is then directly proportional to the amount of samples recorded.

C. Regression Mapping

The righthand plot of Fig. 4 shows an example data set with 600 sample points. As can be seen, the relation between input variables and system output is clearly non-linear. Hence, in order to create a correct mapping, we need to use a non-linear regression method. We decided to use Kernel Ridge Regression with an RBF-kernel, as it is a fairly simple regression method that directly incorporates input data without the need to choose a regression model [17], as would be required e.g. for a Multilayer Perceptron. The only tunable parameter is the width λ of the RBF-kernel. Choosing a high value increases the influence of distant samples, while choosing a low value only includes samples in close vicinity. This parameter is easily tuned by means of cross validation, where the data set is split into a number of subsets, where one set functions as test set and the others as training set. All possible combinations are then tested over a grid of λ -values and the value with the lowest test error is chosen.

Fig. 7 depicts the regression error of the KRR regression for the recorded data set with 600 samples as well as a separate test set with 400 samples.

V. EXPERIMENTS

The mapping generated in Sec. IV has been tested experimentally for planning aperiodic motions on real hardware. The robotic leg consists of spring mechanisms with a constant stiffness $K \approx 2.75 \text{ Nm/rad}$, which are serially connected to position controlled servo units. The first series elastic actuator (SEA) is directly connected to the thigh link, the second SEA is connected to the knee link via belt drives. The total robot size is 25 cm, the hip height when fully stretched is 15.5 cm. The total weight of the leg is about 500 g.

In the experiments we evaluated different types of sequences and scenarios. First, we evaluated the accuracy by commanding linearly increasing jump width, with every width being repeated several times. Second, we tested arbitrary sequences with randomly sampled targets. This is closest to the jump sequences occurring during exploration. Additionally, we sampled targets randomly but limited the increase in jump width so that several jumps are needed to change the jump width from one target to another. An example for this is depicted in Fig. 8 (top). Finally, we experimented with hopping on stairs of different heights and widths. For this we placed a parcours of stairs with heights up to 2 cm. Due to a lack of a vision system the position of each step was taught to the system beforehand³. The system is then programmed to approach the stairs with a fixed step width and then adjust the width dynamically so that the leg lands directly in front of a step and consecutively executes a wide jump. Since every new jump is based on the currently measured state, errors do not accumulate as for offline planned motions. A plot for this is shown in Fig. 8 (bottom).

³in fact, the next step is to replace the hard-coded parcours with an unknown one and determine the position of the stairs by vision. The challenge therefore is on the vision system, while the planning problem is solved by this paper.

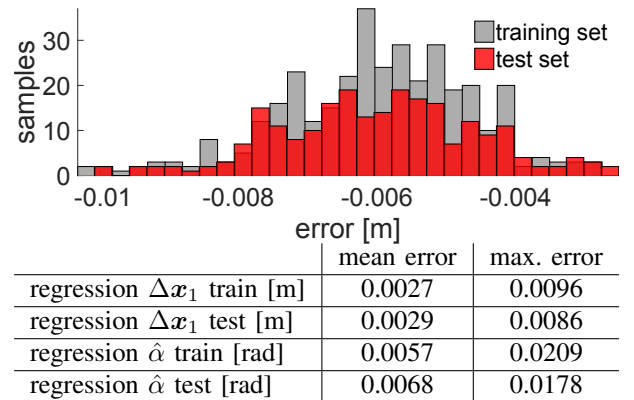


Fig. 7. Regression error for the KRR evaluated with the training data set and test data set. The histogram shows the error for the regression of the base data with Δx_1 as output. The distribution of the histogram for both data sets roughly resembles the Gaussian distribution, which means that the errors are distributed normally. The table also lists the errors for the regression with $\hat{\alpha}$ as target, which is the inverse mapping used in the experiments. Train and test errors are both times close together with train error a bit lower. This allows to conclude that no overfitting is present.

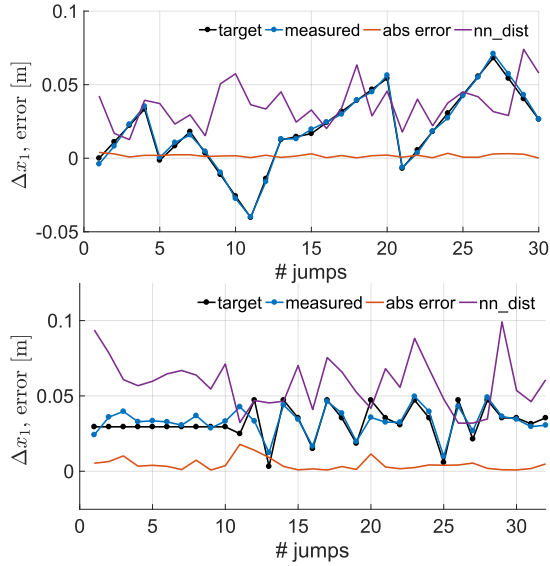


Fig. 8. Error plot for test sequences of the limited random case (top) and the targets during a stairs hopping experiment (bottom). Depicted are the commanded target distance Δx^a and the measured result, as well as the error between both. The purple line nn_dist shows for each jump's set of input parameters an estimation of how well it is covered by the data set in form of the minimal distance to any sample in the data set.

A. Results

As can be seen in the plots in Fig. 8, the robot is able to perform the commanded jump sequences for both cases accurately. A summary of the overall error for each type of experiment is listed in Table II. We recorded about 150 samples for each experiment. The average error in three cases is very similar and is below two millimeters. Relative to the widest jump of all experiments, which was about 6 cm, this is an error of about 3%. However, the maximum error shows significant differences with the biggest error happening during the unrestricted random jumps, where the biggest discrepancies between touchdown velocity and desired step width occur. The best performance in this regard was achieved with the limited random targets, where the conditions are very stable.

VI. CONCLUSION

The main contribution of the paper is exploring the opportunities offered by the concept of a switching based bang-bang controller with discrete parameterization per jump. Despite using serial elastic actuation, which is naturally a complex control task, our methodology allows to plan parameters for targeted jumping without use of a dynamics

model and complex offline calculations. The experiments demonstrate that for different regression algorithms the system is able of performing highly accurate aperiodic hopping maneuvers with online planned parameters. It takes roughly half an hour of time to generate sufficient data to adapt the parameter mapping to changes of the system. Our next steps are the extension to multilegged systems where deeper analysis of the experimental data for better feature extraction is necessary. With higher dimensionality in such systems, the focus also lies on better ways of sample generation to cope with the increased complexity.

REFERENCES

- [1] R. M. Alexander, "Three uses for springs in legged locomotion," *International Journal of Robotics Research*, vol. 9, no. 2, pp. 53–61, 1990.
- [2] G. A. Pratt and M. M. Williamson, "Series elastic actuators," in *Intelligent Robots and Systems 95: Human Robot Interaction and Cooperative Robots*, *Proceedings. 1995 IEEE/RSJ International Conference on*, vol. 1. IEEE, 1995, pp. 399–406.
- [3] D. J. Braun, M. Howard, and S. Vijayakumar, "Exploiting variable stiffness in explosive movement tasks," in *Robotics: Science and Systems*, 2011.
- [4] S. Haddadin, F. Huber, and A. Albu-Schäffer, "Optimal control for exploiting the natural dynamics of variable stiffness robots," in *IEEE Int. Conf. on Robotics and Automation*, 2012.
- [5] M. H. Raibert, *Legged Robots That Balance*. The MIT Press, 1986.
- [6] J. E. Pratt, "Exploiting inherent robustness and natural dynamics in the control of bipedal walking robots," DTIC Document, Tech. Rep., 2000.
- [7] H. Geyer, A. Seyfarth, and R. Blickhan, "Compliant leg behavior explains basic dynamics of walking and running," *Proc. R. Soc. B*, vol. 273, no. 1603, p. 2861–2867, 2006.
- [8] J. Rummel, Y. Blum, H. M. Maus, C. Rode, and A. Seyfarth, "Stable and robust walking with compliant legs," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 5250–5255.
- [9] J. W. Hurst, *The role and implementation of compliance in legged locomotion*. ProQuest, 2008.
- [10] K. Sreenath, H.-W. Park, and J. W. Grizzle, "Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on MABEL," *The International Journal of Robotics Research*, 2013.
- [11] P. Fankhauser, M. Hutter, C. Gehring, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, "Reinforcement learning of single legged locomotion," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 188–193.
- [12] J. Hwangbo, C. Gehring, H. Sommer, R. Siegwart, and J. Buchli, "Rock* — efficient black-box optimization for policy learning," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2014, pp. 535–540.
- [13] D. Lakatos, D. Seidel, W. Friedl, and A. Albu-Schäffer, "Targeted jumping of compliantly actuated hoppers based on discrete planning and switching control," in *International Conference on Intelligent Robots and Systems*. IEEE, 2015, pp. 5802–5808.
- [14] M. Raibert and F. C. Wimberly, "Tabular control of balance in a dynamic legged system," *IEEE Transactions on Systems, Man and Cybernetics*, no. 2, pp. 334–339, 1984.
- [15] R. A. Brooks, "Artificial life and real robots," in *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life*. MIT press, 1992, p. 3.
- [16] D. Lakatos, F. Petit, and A. Albu-Schäffer, "Nonlinear oscillations for cyclic movements in human and robotic arms," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 865–879, 2014.
- [17] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

TABLE II
EXPERIMENTAL RESULTS.

	#	mean error	max. error
linear	160	0.0017 (2.8%)	0.0078 (12.8%)
random	150	0.0018 (3.0%)	0.0086 (14.3%)
lim. random	150	0.0019 (3.2%)	0.0058 (9.5%)
all	460	0.0018 (3.0%)	0.0086 (14.3%)